1. Which of the following fills in the blank so that the code outputs one line but uses a
poor practice?

```
import java.util.*;
public class Cheater {
int count = 0;
public void sneak(Collection<String> coll) {
coll.stream()._______________________;
}
public static void main(String[] args) {
Cheater c = new Cheater();
c.sneak(Arrays.asList("weasel"));
}
}
```

A. `peek(System.out::println)`
B. `peek(System.out::println).findFirst()`
C. `peek(r -> System.out.println(r)).findFirst()`
D. `peek(r -> {count++; System.out.println(r); }).findFirst()`

2. Which can fill in the blank to have the code print `true`?

```
Stream<Integer> stream = Stream.iterate(1, i > i+1);
boolean b = stream.____________(i -> i > 5);
System.out.println(b);
```

A. `anyMatch`
B. `allMatch`
C. `noneMatch`
D. None of the above

3. How many of the following can fill in the blank to have the code print `44`?

```
Stream<String> stream = Stream.of("base", "ball");
stream.____________(s -> s.length()).forEach(System.out::print);
```

I. `map`
II. `mapToInt`
III. `mapToObject`
A. None
B. One

C. Two
D. Three

4. Which of these stream pipeline operations takes a `Predicate` as a parameter and
returns an `Optional`?
A. `anyMatch()`
B. `filter()`
C. `findAny()`
D. None of the above

5. What is the result of the following?

```
List<Double> list = new ArrayList<>();
list.add(5.4);
list.add(1.2);
Optional<Double> opt = list.stream().sorted().findFirst();
System.out.println(opt.get() + ″ ″ + list.get(0));
```

A. `1.2 1.2`
B. `1.2 5.4`
C. `5.4 5.4`
D. None of the above

6. How many of these collectors can fill in the blank to make this code compile?

```
Stream<Character> chars = Stream.of(
'o', 'b', 's', 't', 'a', 'c', 'l', 'e');
chars.map(c -> c).collect(Collectors.___________ );
```

I. `toArrayList()`
II. `toList()`
III. `toMap()`
A. None
B. One
C. Two

D. Three

7. What does the following output?

```
import java.util.*;
public class MapOfMaps {
public static void main(String[] args) {
Map<Integer, Integer> map = new HashMap<>();
map.put(9, 3);
Map<Integer, Integer> result = map.stream().map((k,v) > (v,k));
System.out.println(result.keySet().iterator().next());
}
}
```

A. `3`
B. `9`
C. The code does not compile.
D. The code compiles but throws an exception at runtime.

8. Which of the following creates an `Optional` that returns `true` when calling
`opt.isPresent()`?
I. `Optional<String> opt = Optional.empty();`
II. `Optional<String> opt = Optional.of(null);`
III. `Optional<String> opt = Optional.ofNullable(null);`
A. I
B. I and II
C. I and III
D. None of the above

9. What is the output of the following?

```
Stream<String> s = Stream.of("speak", "bark", "meow", "growl");
BinaryOperator<String> merge = (a, b) -> a;
Map<Integer, String> map = s.collect(Collectors.toMap(String::length,
k -> k, merge));
System.out.println(map.size() + " " + map.get(4));
```

A. `2 bark`
B. `2 meow`
C. `4 bark`
D. None of the above

10. What is the output of the following?

```
package reader;
import java.util.stream.*;

public class Books {
public static void main(String[] args) {
IntegerStream pages = IntegerStream.of(200, 300);
IntegerSummaryStatistics stats = pages.summaryStatistics();
long total = stats.getSum();
long count = stats.getCount();
System.out.println(total + "-" + count);
}
}
```

A. `500-0`
B. `500-2`
C. The code does not compile.
D. The code compiles but throws an exception at runtime.

11. What is true of the following code?

```
Stream<Character> stream = Stream.of('c', 'b', 'a'); // z1
stream.sorted().findAny().ifPresent(System.out::println); // z2
```

A. It is guaranteed to print the single character `a`.
B. It can print any single character of `a`, `b`, or `c`.
C. It does not compile because of line `z1`.
D. It does not compile because of line `z2`.

12. Suppose you have a stream pipeline where all the elements are of type `String`. Which
of the following can be passed to the intermediate operation `sorted()`?
A. `(s,t) -> s.length() - t.length()`
B. `String::isEmpty`

C. Both of these
D. Neither of these

13. Fill in the blanks so that both methods produce the same output for all inputs.

```
private static void longer(Optional<Boolean> opt) {
if (opt.___________())
System.out.println("run: " + opt.get());
}
private static void shorter(Optional<Boolean> opt) {
opt.map(x -> "run: " + x).___________(System.out::println);
}
```

A. `isNotNull`, `isPresent`
B. `ifPresent`, `isPresent`
C. `isPresent`, `forEach`
D. `isPresent`, `ifPresent`

14. What does the following output?

```
Set<String> set = new HashSet<>();
set.add("tire-");
List<String> list = new LinkedList<>();
Deque<String> queue = new ArrayDeque<>();
queue.push("wheel-");
Stream.of(set, list, queue)
.flatMap(x -> x.stream())
.forEach(System.out::print);
```

A. `[tire-][wheel-]`
B. `tire-wheel-`
C. None of the above.
D. The code does not compile.

15. What is the output of the following?

```
Stream<String> s = Stream.of("over the river",
"through the woods",
"to grandmother's house we go");
s.filter(n -> n.startsWith("t"))
```

```
.sorted(Comparator::reverseOrder)
.findFirst()
.ifPresent(System.out::println);
```
A. `over the river`
B. `through the woods`
C. `to grandmother's house we go`
D. None of the above

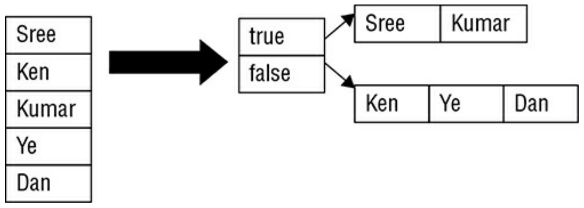
16. Which fills in the blank so the code is guaranteed to print `1`?
```
Stream<Integer> stream = Stream.of(1, 2, 3);
System.out.println(stream.___________);
```
A. `findAny()`
B. `first()`
C. `min()`
D. None of the above

17. Which of the following can be the type for `x`?
```
private static void spot(___________ x) {
x.filter(y -> ! y.isEmpty())
.map(y -> 8)
.ifPresent(System.out::println);
}
```
I. `List<String>`
II. `Optional<Collection>`
III. `Optional<String>`
IV. `Stream<Collection>`
A. I
B. IV
C. II and III
D. II and IV

18. Which can fill in the blank to have the code print `true`?
```
Stream<Integer> stream = Stream.iterate(1, i > i);
boolean b = stream.___________(i -> i > 5);
System.out.println(b);
```

A. `anyMatch`
B. `allMatch`
C. `noneMatch`
D. None of the above

19. What collector turns the stream at left to the `Map` at right?



A. `grouping()`
B. `groupingBy()`
C. `partitioning()`
D. `partitioningBy()`

20. Which fills in the blank for this code to print `667788`?

```
IntStream ints = IntStream.empty();
IntStream moreInts = IntStream.of(66, 77, 88);
Stream.of(ints, moreInts).___________(x -> x).forEach(System.out::print);
```

A. `flatMap`
B. `flatMapToInt`
C. `map`
D. None of the above